2016 Fourth International Conference on Information and Communication Technologies (ICoICT)

Performance Analysis of Container-based Hadoop Cluster : OpenVZ and LXC

Rizki Rizki
School of Computing
Telkom University
Buah Batu, Bandung 40257
rizkidoank@students.telkomuniversity.ac.id

Andrian Rakhmatsyah ST., MT.
School of Computing
Telkom University
Buah Batu, Bandung 40257
kangandrian@telkomuniversity.ac.id

M. Arief Nugroho ST., MT.
School of Computing
Telkom University
Buah Batu, Bandung 40257
arif.nugroho@telkomuniversity.ac.id

***Abstract*—Hadoop is an open-source software framework that commonly used for distributed processing and distributed storage in large datasets (big data). It perform processing and stored the data across all nodes in cluster. But, the deployment and operational costs of physical cluster are costly in some reasons. Physical cluster needs high energy and rigid in manageability. As the solution, virtualization technology used to get more flexibility and reduce costs. But, hypervisor-based virtualization gives more I/O performance overhead compared to bare-metal server. Nowadays, container-based virtualization are used to obtain lower performance overhead. Open Virtuozzo (OpenVZ) and Linux Container (LXC) are the examples of container-based virtualization. Implementation of container-based virtualization gives near native performances.**

I. Introduction

Hadoop is a framework for distributed processing and distributed storage in very large datasets [1], its designed to work on physical nodes. But, the deployment and maintenances costs are costly. Beside that, misconfiguration of clusters become the main cause in hadoop cluster break issues. It took up to $\pm 35\%$, misconfiguration includes hadoop configuration and OS resources limits [2]. Physical cluster are rigid in configurations related. As the solution, virtualization used to get more flexibility and reduce costs. But, virtualization give more I/O performance overhead that caused by the hypervisor itself like KVM, vSphere, or Xen [3].

In the other side, the development of virtualization is keep going. Nowadays, container-based virtualization used to obtain more performance. Container-based virtualization give near native performances. It works on operating system level, sharing the same kernel as host [4].

Open Virtuozzo (OpenVZ) is a container-based virtualization that works on operating-system level with modified kernel on Linux [5]. Linux Container (LXC) is *userspace interface* for containers feature in newer Linux kernel, it makes the system possible to deploy one or more containers by host shared kernel [6].

This paper focused on container-based virtualization implementation on virtualized hadoop cluster and performance analysis of container-based hadoop cluster on OpenVZ and LXC by its I/O throughput and average time executions on hadoop MapReduce.

OpenVZ is one of the most commonly used container-based in production. It had been stable for a long time. LXC is still in development, its already come in Linux newer kernel.Its used as the backend on early release of docker.

II. MapReduce

MapReduce is a programming model to processed distributed large datasets through distributed filesystem (e.g Hadoop Distributed Filesystem, Google Filesystem) in parallel in distributed environment [7]. There are two main functions in MapReduce programming model : map function, and reduce function. There are many engines that use MapReduce programming model (e.g., Hadoop [1], Spark [8], Tez [9]). This work will use default engine for Apache Hadoop named Hadoop MapReduce.

III. Apache Hadoop

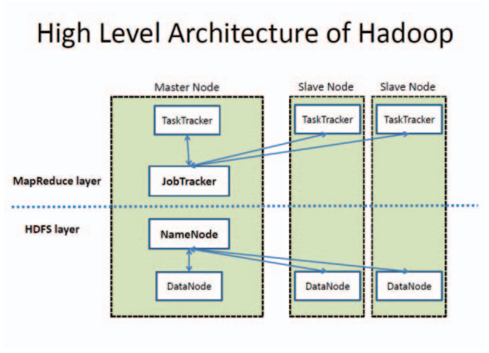

Fig. 1. Hadoop minimal architecture

Hadoop is an open-source framework based on Java that used for distributed storage and distributed processing in very large datasets [1]. Hadoop designed to work in physical nodes and gives high scalability, flexibility, and fault tolerant system. Hadoop project includes *Hadoop Common, Hadoop Distributed Filesystem, Hadoop YARN, and Hadoop MapReduce*. Figure 1 shown the minimal architecture of hadoop with only HDFS and YARN services on it. In real world, hadoop ecosystem is not only HDFS and YARN. But, other services like HBase, Hive, Storm, Spark are contributed, too.

ISBN: 978-1-4673-9879-4 (c) 2016 IEEE

A. Hadoop Distributed Filesystem

Hadoop Distributed Filesystem (HDFS) is a distributed filesystem that provides scalable, reliable data storage, and high throughput [10]. HDFS includes two components: NameNode, DataNode. NameNode used to store the metadata of stored data in DataNodes. DataNode used to store the data for being processed. The data split into blocks with defined size, 128MB by default. And then, the blocks distributed across the nodes and replicated according to the replication factor. The replication on HDFS gives more availability in data integrity. When some data is corrupted, it may be saved by its other replications. NameNode and DataNode should not in same host, and some references recommends to use 3 DataNodes in minimal.

B. Hadoop Yet Another Resources Negotiator (YARN)

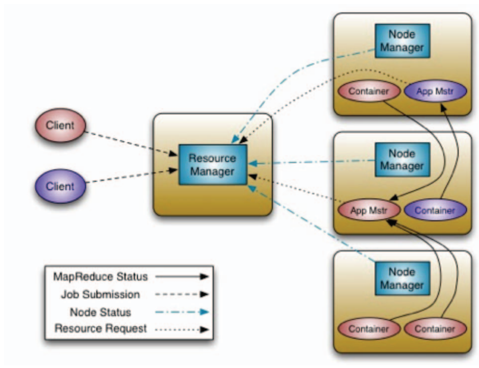

Fig. 2. YARN architecture

The Figure 2 shown the architecture of YARN. Hadoop YARN is the next release of Hadoop MapReduce v1 (MRv1). In MRv2, resources management done by YARN, while data processing handled by MapReduce or any other engines. MapReduce includes two components : JobTracker and TaskTracker. JobTracker is a service for cluster job management. JobTracker accept requests from client, and then executed by TaskTracker. TaskTracker will inform JobTracker if the job done or failed [11]. Each job processed in batch, client may submit the jobs and waiting until it done. Job monitoring can be shown on HistoryServer in web interface. In HistoryServer, client able to see every jobs state (running, failed, success, pending).

IV. Container-based Virtualization

Container-based virtualization is a virtualization method that works on operating-system level by sharing kernel host to create one or more instances / containers [12]. This virtualization gives near native performance, lower in performance overhead compared by full-virtualization or paravirtualization. It works without hypervisor / virtualization applications layer. In this work, OpenVZ and LXC used in our experiment.

As can be seen on Figure 3, most hypervisor-based virtualization provides Virtual Machine Monitor as the abstraction layer of VM [12]. It makes the guests VM to run on isolated

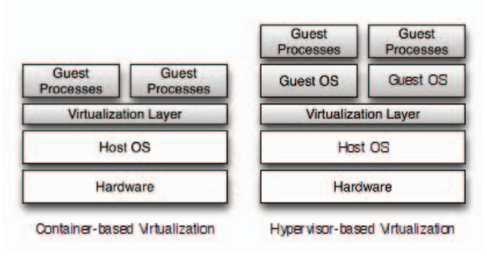

Fig. 3. Container-based and Hypervisor-based architecture

environment and have its own kernel. While in container-based virtualization, the instances run directly as guests processes. By this reason, every instances on container-based virtualization work on the same kernel as host. The isolation in LXC done by kernel namespaces [12].

A. Open Virtuozzo (OpenVZ)

OpenVZ is a container-based virtualization established since 2006. The resources isolation are stable and currently used in production on some hosting providers. But, disk I/O limit is not possible in OpenVZ. OpenVZ currently use modified linux kernel 2.6 (vzkernel). OpenVZ use vzctl as the container management tools [5]. User Beancounters used in OpenVZ as the resources controller and management. By default, OpenVZ restricts the access to physical devices.

B. Linux Containers (LXC)

LXC is container-based virtualization that currently in development. It works by using linux kernel features, cgroups and namespaces Cgroups was used as the resources management in LXC, its includes : core count, memory limit, disk I/O limit, etc. Namespaces used to isolate the containers in process, its includes : PID (child PID / nested PID), filesystem (chroot), etc. In this work, LXC v1 used on experiment. Since the current version of LXC have some bugs in some OS, Proxmox VE4 was used to get more stability. LXC gives more access to physical devices, in example accessing host GPU.

V. Experiments

Our work were conducted by implementations of container-based virtualization, in this case OpenVZ and LXC. The experiments are benchmarking the virtualized clusters to measure the performances in terms of I/O performance overhead, and average time execution of mapreduce job.

Our testbed use one host for each clusters. The hosts have specifications as following : Intel i7 Quad Core, 8GB DDR3 RAM, 1 x 1TB HDD, and Ethernet NIC. Each host contained three nodes as the cluster nodes, resources configurations sets as following : two core, 2GB RAM, 100GB disk space, and one virtual ethernet. Hosts kernel requirement are differ each other. In this experiments, OpenVZ stable used on top of Proxmox VE 3.4 which use kernel 2.6, while LXC used on top of Proxmox VE 4 which use kernel 3.1. Any container management are done by Proxmox VE web-based management. Each hosts use same I/O scheduler, in this case CFQ scheduler.

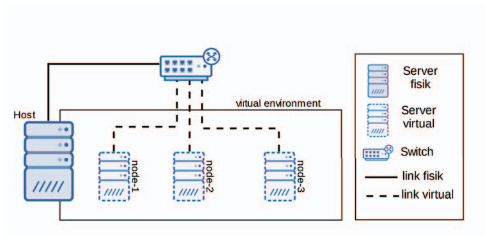

Fig. 4. Cluster architecture

Hadoop needs to be configured to make the cluster. We set up the base configurations for HDFS and YARN by Cloudera CDH 5 [13]. Each cluster contained 3 nodes, 1 node as the NameNode and JobTracker, the others 2 nodes as the DataNode, TaskTracker, and HistoryServer. Java Heap size set to 1736MB. Our architecture shown as Figure

VI. Benchmarks

A. Benchmarks Scenario

To measure the clusters performances, we used applications provided by hadoop packages. TestDFSIO used to measure the I/O performances of the cluster, WordCount used to measure the average time execution. TestDFSIO commonly used by hadoop users for identifying performances bottlenecks in networks, operating system, and also configurations of HDFS [14]. We ran tests by writing and reading $N \times 64\text{MB}, N = 1, 2, 4, 8, 16, 32$. In this experiment, default replication factor value was used. The default value is 3, it means each tests $3 \times (N \times 64MB)$ are transferred and distributed across DataNodes. MapReduce average time execution tests done by ran Syslog Priority Parser and Counter with syslog CSV data from Telkom University Information System Department. Syslog data size is 4989521886 in Bytes (4.7GB).

B. HDFS Benchmark

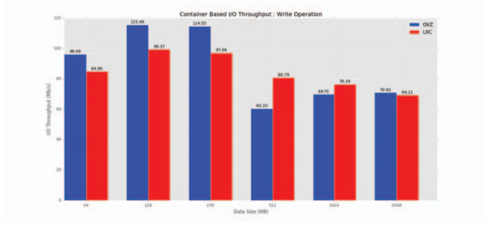

Fig. 5. HDFS I/O Performance : write operation

Figures 5 and 6 shown the benchmark results for both write operations and read operations on LXC and OpenVZ. Figure 5 shows that OpenVZ got better results in write operations rather than LXC with CFQ scheduler on both. The throughput is quite far each other, but started from 512MB OpenVZ throughput dropped and become comparative with LXC in the next tests. As shown in Figure 6, LXC throughput quite better compared to OpenVZ, the difference almost ±200Mb/s. but, LXC got significant drop while in 1GB and 2GB tests. The difference is quite far between OpenVZ and LXC in 1GB and 2GB tests.

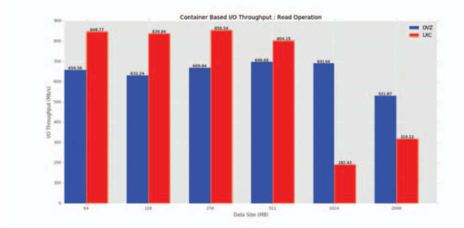

Fig. 6. HDFS I/O Performance : read operation

C. MapReduce Execution Time

MapReduce execution time are tested with rsyslog analysis to get and counts syslog priority in a dataset. Dataset taken from University Information System Department. In Table I, shown the details of the dataset. There are 9 fields in the dataset : *hostname, fromhost-ip, timereported, timegenerated, msg, syslogfacility, syslogseverity, programname, syslogpriority-text*. The purpose of the program is to counts *syslogpriority-text* in the dataset.

TABLE I
Dataset Details

Size (Bytes)	4989521886
Num of Records	50235249
Num of Fields	9
Filetype	CSV

Program written in Python and Hadoop Streaming to get running on cluster. As can be observed in Figure 7, OpenVZ gives more better performances regarding the execution time of MapReduce Jobs.

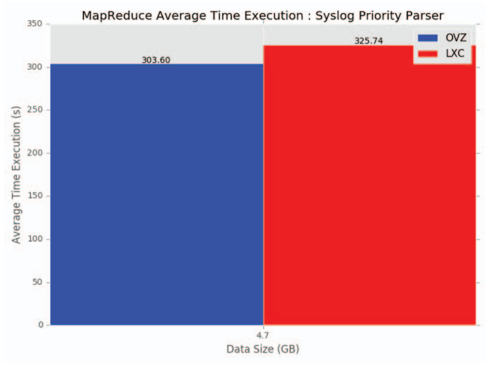

Fig. 7. Syslog Priority Parser average time executions on container-based virtualization

VII. Conclusion

Container-based hadoop cluster gives near native performances because of the lightweight virtualization on operating system level. In our experiment with 3 nodes on Proxmox VE using CFQ scheduler. The results shown that OpenVZ is more stable in current configurations. While in Syslog Priority Parser benchmarks as can be observed on Figure 7, the results are comparative. But, OpenVZ gives more faster running time.

As future work, we can observes the container-based cluster performance in some different scheduler in operating-system level e.g., *noop,deadline, etc* as well as on the hadoop scheduler itself to fit the best configurations for container-based cluster.

References

[1] A. Hadoop, "Apache hadoop," 2011. [Online]. Available: https://hadoop.apache.org/

[2] A. Rabkin and R. Katz, "How hadoop clusters break," *Software, IEEE*, vol. 30, no. 4, pp. 88–94, July 2013.

[3] N. Regola and J.-C. Ducom, "Recommendations for virtualization technologies in high performance computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 409–416.

[4] P. Magalhaes Vasconcelos and G. Azevedo de Araujo Freitas, "Performance analysis of hadoop mapreduce on an opennebula cloud with kvm and openvz virtualizations," in *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*, Dec 2014, pp. 471–476.

[5] OpenVZ, "Openvz linux containers," 2015. [Online]. Available: http://openvz.org

[6] LXC, "Lxc - linux containers," 2015. [Online]. Available: https://linuxcontainers.org/lxc/introduction/

[7] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[8] A. Spark, "Lightning-fast cluster computing,apache spark, 2015."

[9] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino, "Apache tez: A unifying framework for modeling and building data processing applications," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015, pp. 1357–1369. [Online]. Available: http://doi.acm.org/10.1145/2723372.2742790

[10] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating mapreduce on virtual machines: The hadoop case," in *Cloud Computing*, ser. Lecture Notes in Computer Science, M. Jaatun, G. Zhao, and C. Rong, Eds. Springer Berlin Heidelberg, 2009, vol. 5931, pp. 519–528. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10665-1_47

[11] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 5.

[12] M. Xavier, M. Neves, F. Rossi, T. Ferreto, T. Lange, and C. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, Feb 2013, pp. 233–240.

[13] I. Cloudera, "Cdh proven, enterprise-ready hadoop distribution–100% open source," 2012.

[14] M. Gomes Xavier, M. Veiga Neves, and C. Fonticielha de Rose, "A performance comparison of container-based virtualization systems for mapreduce clusters," in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, Feb 2014, pp. 299–306.

[15] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[16] T. Ivanov, R. Zicari, and A. Buchmann, "Benchmarking virtualized hadoop clusters," in *Big Data Benchmarking*, ser. Lecture Notes in Computer Science, T. Rabl, K. Sachs, M. Poess, C. Baru, and H.-A. Jacobson, Eds. Springer International Publishing, 2015, vol. 8991, pp. 87–98. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-20233-4_9